Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Jakob Šalej

Mobilni nadzor transporta živil

diplomsko delo

univerzitetni študijski program prve stopnje
računalništvo in informatika

Mentor: doc. dr. Mira Trebar

Ljubljana, 2017

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Sodoben transport proizvodov je povezan z različnimi sistemi in tehnologijami za zbiranje in obdelavo podatkov. Kandidat naj v diplomskem delu zasnuje rešitev, ki z uporabo senzorskih meritev pametnega telefona in drugih spletnih storitev omogoča nadzor prevoza za eno ali več naročil hkrati. Implementira naj mobilno aplikacijo za operacijski sistem Android, ki pridobiva podatke o trenutni lokaciji, temperaturi in vlagi ter dostopa do podatkovne baze na strežniku z uporabo RESTful storitev. Rezultate testiranja naj predstavi na enostavnem primeru simuliranega prevoza vse od sprejema naročila do zaključka transporta.

Iskrena zahvala mentorici doc. dr. Miri Trebar za potrpežljivost, spodbudo in usmerjanje pri pisanju diplomskega dela. Zahvaljujem se tudi svoji družini in dekletu Maji za vso podporo v času študija.

Maji, za vse spečene pice v času pisanja.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
ADT	Android development tools	razvojna orodja za Android
API	application programming interface	aplikacijski programski vmesnik
GPS	global positioning system	globalni pozicijski sistem
HTML	hypertext markup language	hiper tekstovni označevalni jezik
IAP	in-app purchases	nakupi znotraj aplikacije
IDE	integrated development environment	integrirano razvojno okolje
JSON	javascript object notation	javascript objektna notacija
OS	operating system	operacijski sistem
PaaS	platform as a service	platforma kot storitev
QR	quick response	hitri odgovor
RAM	random-access memory	bralno-pisalni pomnilnik
REST	representational state transfer	predstavitveni prenos stanja
SOAP	simple object access protocol	protokol za spletne storitve
URL	uniform resource locator	enotni identifikator vira
VCS	version control system	sistem za nadzorovanje verzij
WS	web service	spletna storitev
WSDL	web services description language	opisni jezik spletnih storitev
XML	extensible markup language	razširljivi označevalni jezik

Povzetek

Naslov: Mobilni nadzor transporta živil

Avtor: Jakob Šalej

Kljub velikemu napredku na področju transporta se še danes ogromno živil zavrže, preden sploh pridejo do potrošnikov. Eden od glavnih razlogov za to je neustrezen nadzor transporta. S porastom potreb po svežih izdelkih na eni in zamrznjenih na drugi strani, je vedno bolj pomembno zagotavljanje pravih pogojev, predvsem ustrezne temperature in vlage. S potrebo po večjem nadzoru pa se poveča tudi odgovornost tistih, ki skrbijo za transport. V diplomski nalogi smo zasnovali mobilni nadzor transporta živil z uporabo ene same naprave, ki jo ima dandanes že skoraj vsak vedno pri sebi - pametnega mobilnega telefona. GPS lokacija naprave bo uporabljena za pridobivanje podatkov o temperaturi in vlagi z uporabo spletnih storitev. Mobilna aplikacija, ki bo tekla na operacijskem sistemu Android, bo omogočila spremljanje in shranjevanje vseh podatkov o posameznem prevozu, analizo kritičnih točk v verigi in sporočanje neustreznih dogodkov. Pri razvoju smo se osredotočili na implementacijo preglednega uporabniškega vmesnika, ki uporabniku nenehno podaja relevantne informacije o stanju transporta in v realnem času opozarja na odstopanja od priporočenih pogojev.

Ključne besede: transport, živila, nadzor, Android, mobilna, aplikacija, temperatura.

Abstract

Title: Mobile monitoring of food transport

Author: Jakob Šalej

Despite the great progress made in food transportation, goods are still being thrown away before they even reach consumers. One of the main reasons for that is inadequate monitoring of transports. With growing demand for both fresh and frozen products, it is becoming increasingly important to ensure adequate conditions, mainly appropriate temperature and humidity. This need for better monitoring puts more pressure on those responsible for transport. We designed a mobile monitoring of food transport using a single device, which is nowadays within everybody's reach - a smartphone. GPS location of the device will be used to retrieve temperature and humidity data using web services. A mobile application running on Android operating system will allow users to monitor and store data of each transport, analyze critical points in transport, and report abnormal events. During development, we focused on implementation of a transparent user interface that gives the user a relevant information about the transport and issues real-time alerts, when conditions are sub-optimal.

Keywords: transport, foods, monitoring, Android, mobile, application, temperature.

Poglavje 1

Uvod

Ko je pred nekaj več kot 200 tisoč leti Jurak pogledal v nebo, je bilo v njegovih očeh moč opaziti zmagoslaven blisk. Dolgi, temni lasje so mu prekrili zgornji del porjavelega vratu, spredaj pa je brada, nekoliko posvetljena od sonca, ponosno strmela v nebo. Po dolini se je razlegel globok krik veselja. Ne zgodi se pogosto, da bi Homo sapiens lahko pričakoval obilno večerjo, a danes jo lahko. Le še odnesti mora meso do svoje votline...

Sonce je že zahajalo v daljavi in Jurak rame, čez katero si je vrgel svoj ulov, sploh ni več čutil. Neprestano odganjanje mrčesa, ki ga je privabil vonj toplega mesa, je bilo še dodatno utrujajoče. Spraševal se je, zakaj ni na lov vzel še koga. Potem bi breme lahko nosil on. Je pa res, da bi potem verjetno zahteval kos mesa, je še pomislil in nadaljeval s potjo. Vse bolj se je mračilo. Do trenutka, ko je lahko končno odložil tovor v svoji jami, so bile na nebu že dolgo le še zvezde in luna.

Kot tisoči pred njim, je Jurak opravil le še enega od transportov hrane. Transport, ki mu omogoča, da bo naslednje štiri dni preždel na varnem v svoji jami, stran od divjih predatorjev, ki ti lahko v vsakem trenutku odtrgajo glavo. Transport, ki mu omogoča preživetje. Spet je bil dobre volje.[1]

[1] Zgodba je izmišljena

Danes je transport zelo razširjen in poteka po cesti, vodi in zraku. Uporabljajo se različna prevozna sredstva, od ladij, avtov, tovornjakov in vlakov, pa vse do letal. A čeprav so prevozna sredstva druga, pa so živila - tako kot nekoč - ena prevladujočih dobrin transporta.

Pri transportu živil je zelo pomembno, da je tovor kar najhitreje prepeljan z lokacije A do lokacije B. Še pomembneje pa je, da ima prevozniško podjetje ustrezen nadzor nad celotnim transportom. Kako to storiti lažje, kot pa z uporabo tiste naprave, ki jo imamo vedno pri sebi - pametnega telefona? Temu se je v zadnjih letih zmogljivost izredno povečala in skupaj z velikim številom različnih senzorjev, ki jih premorejo, so mobilne naprave idealna platforma za različne rešitve.

Odločili smo se, da v okviru diplomske naloge izdelamo **mobilni nadzor transporta živil**, ki bo podjetju in njegovim voznikom v pomoč pri spremljanju dogajanja tako med, kakor tudi po končanem transportu živil. Cilj diplomske naloge je tako izdelati mobilno aplikacijo za operacijski sistem Android, ki bo uporabnikom (prevozniškemu podjetju) omogočila enostavnejši in preglednejši nadzor nad transportom živil. Mobilna aplikacija bo delovala od trenutka, ko voznik prevzame naročilo, pa do trenutka, ko ga pripelje na cilj. Med potekom se bodo zajemali podatki o lokaciji naprave (GPS koordinate). Z uporabo spletne storitve bodo na voljo meritve temperature in vlažnosti ozračja za to lokacijo. Z zapisom vseh dogajanj pri vsakem vozniku bo prevozniško podjetje imelo relevantne informacije o poteku transporta, za katerega je voznik odgovoren.

Po uvodu in seznanitvi s tematiko diplomskega dela, sledi v drugem poglavju podrobnejši pregled področja nadzora transporta. Začeli bomo z opisom problema, prešli na zasnovo rešitve in končali s praktičnim primerom uporabe.

V tretjem poglavju bomo naredili kratek pregled glavnih tehnologij, uporabljenih pri izdelavi rešitve. Podrobneje si bomo pogledali operacijski sistem Android, saj bo to gostujoči operacijski sistem (OS) naše mobilne aplikacije, hkrati pa nam bo omogočil uporabo dveh senzorjev naprave: GPS senzorja, s katerim bomo določali pozicijo naprave, in pa kamere, s katero bomo brali QR kode. Poleg tega se bomo dotaknili še področja spletnih storitev, saj bo ena izmed njih igrala pomembno vlogo v naši aplikaciji.

V četrtem poglavju si bomo ogledali potek razvoja mobilne aplikacije, ki vključuje načrtovanje, implementacijo in testiranje končne rešitve na realnem primeru uporabe. Z izsledki testiranj se bomo v zadnjem poglavju dotaknili uporabne vrednosti naše rešitve, kaj so njene prednosti in kaj slabosti ter kaj bi se v prihodnosti dalo še izboljšati.

Poglavje 2

Nadzor transporta

Transport je ena izmed najpomembnejših dejavnosti v človeškem obstoju. Če gledamo širši pomen besede *transport*, si brez dejavnosti, ki jih ta pojem zajema, življenja danes verjetno sploh ne znamo predstavljati. V angleškem slovarju zanj najdemo naslednjo definicijo: *prenos (ljudi ali dobrin) z ene lokacije na drugo* (ang. *the movement of people or goods from one place to another*). Pomen izhaja iz latinske besede *transportare*; *trans* v dobesednem prevodu pomeni *čez*, *portare* pa *nositi*.[1] V SSKJ pa med drugim najdemo naslednjo definicijo, ki dobro zadane obseg besede *transport* v okviru tega diplomskega dela: *gospodarska dejavnost, ki se ukvarja s prevozom*.[2]

V diplomskem delu bo poudarek na prenosu dobrin, natančneje živil. Čeprav je res, da gre v uvodu za izmišljeno zgodbo o fantu Juraku, pa le-ta temelji na zgodovinskih dejstvih. Skuša nam pokazati, kako pomembno vlogo igra prenos dobrin že od samega začetka razvoja človeka. Kljub temu, da je same dokaze o tem težko najti, pa najzgodnejša najdba, ki direktno dokazuje obstoj transporta, sega kar 2.5 milijona let nazaj v Etiopijo. Prenašalo se je predvsem hrano, vodo, les za ogenj in orodje - stvari, ki so za preživetje ključne [12].

[1] `http://www.dictionary.com/browse/transport`

[2] `http://bos.zrc-sazu.si/cgi/a03.exe?name=sskj_testa&expression=transport&hs=1`

Potrebe po transportu živil so ogromne. Po podatkih strani *Eurostat* so leta 2015 živila in živilski proizvodi predstavljali le nekaj manj kot 30% celotnega prevoznega trga v EU v tonah-kilometer (enota transporta dobrin, ki predstavlja prevoz ene tone blaga z uporabo danega prevoznega sredstva na razdalji enega kilometra) [5].

Žalosti dejstvo, da po nekaterih podatkih približno 33% pridelane hrane širom sveta ni nikoli uporabljene [10]. Velik del tega predstavljajo živila, ki se pokvarijo zaradi neprimernega ravnanja in neustreznega nadzora nad transportom. Tako bi samo s spremljanjem temperature v realnem času in takojšnjem sporočanju morebitnih odstopanj - ta omogočijo nadzorniku transporta, da se na težavo ustrezno in hitro odzove - lahko dosegli opazno zmanjšanje količine živil, ki jih je potrebno po koncu transporta zavreči [11].

2.1 Opis problema

Za prevozniško podjetje je dober nadzor transportov bistvenega pomena. Vedno morajo biti na voljo informacije o tem, kateri voznik je opravil določeno naročilo, kdaj ga je opravil, kdaj je začel s transportom, kdaj končal in drugo. Te informacije so nujne za uspešno logistično delovanje podjetja.

Pri naročilu pa ni važno le, da je opravljeno; važna je tudi kakovost storitve. Poleg same hitrosti opravljene storitve (ki je posredno odvisna tudi od uspešnosti logistike podjetja!) je glavno merilo kakovosti stanje tovora po opravljenem transportu. Podjetju nič ne pomaga najhitrejša dostava v regiji, če pa je tovor ob prihodu na cilj poškodovan ali celo neuporaben.

Kako lahko podjetje zagotovi ustrezno stanje tovora po transportu? Z nadzorom transporta, ki vključuje zajemanje meritev vrednosti tistih dejavnikov, ki vplivajo na stanje tovora. Da bi lahko podjetje ustrezno odreagiralo

in preprečilo morebitne težave s tovorom, mora dobivati realne vrednosti teh dejavnikov in v primeru odstopanj prejeti takojšnje obvestilo.

Pri transportu živil je najpomembnejši dejavnik temperatura, saj so živila v splošnem hitreje pokvarljiva. Hlajenje je ena najbolj pogostih metod za upočasnjeno rast bakterij, ki vplivajo na živila. Ustrezen nadzor in upravljanje temperature sta ključna dejavnika pri ohranjanju kakovosti [8].

Z večjo potrebo po nadzoru med samim transportom tako nastopi tudi večja odgovornost za prevozna podjetja, ki morajo zagotoviti, da živila tudi po koncu transporta še vedno ustrezajo varnostnim in kakovostnim standardom. Morebitni odpoklici izdelkov ali težave z njihovo kakovostjo lahko škodujejo imenu podjetja in posledično negativno vplivajo na poslovanje. Sistemi, ki sporočajo in beležijo stanje skozi celoten transport, zmanjšujejo možnost pojavitve zdravju nevarnih in nizkokakovostnih produktov. Ti sistemi niso le orodje, ki pripomore, da živila ustrezajo zahtevanim kakovostnim standardom, temveč so tudi način, kako pridobiti zaupanje strank [9].

2.2 Zasnova rešitve

Zakaj ne bi mogel prevoznik sprejemati naročil in nadzorovati transportov kar z napravo, ki jo ima vedno pri sebi? S svojim pametnim mobilnim telefonom, ki je vedno povezan v svetovni splet in ima veliko število različnih senzorjev? Izdelati želimo mobilno aplikacijo, ki bo omogočala prevozniku enostaven nadzor nad enim ali več transportov in prikaz podatkov v realnem času.

Rešitev bo sestavljena iz dveh delov - strežnika in mobilne aplikacije. Mobilna aplikacija bo dostopala do podatkovne baze na strežniku preko REST (predstavitveni prenos stanja) storitev. Želimo aplikacijo, ki bo morala le

prenesti naročilo s strežnika, nato pa samostojno delovala vse do zaključka prevoza, ko bo posodobljene podatke poslala nazaj na strežnik.

Za vsak transport v izvajanju se bodo k prevozniku shranjevali podatki o lokaciji ter temperaturi in vlagi ozračja. Za pridobivanje lokacije bomo uporabili senzor GPS mobilne naprave. Po koncu bo možna analiza kritičnih točk v transportu, sicer pa bo aplikacija že med samim zajemanjem meritev sporočala neustrezne dogodke.

Poglejmo si potek delovanja od prevzema do oddaje naročila:

- prevozniško podjetje ima N prevoznikov
- prevoznik se na podlagi uporabniškega imena vpiše v aplikacijo
- prevoznik prebere QR kodo svojega vozila
- prevoznik ob prevzemu naročila prebere QR kodo na dobavnici
- na podlagi te kode se iz strežnika podjetja na mobilno napravo prenese ustrezen dokument, ki vsebuje podatke o transportu
- ob začetku transporta se začne tudi zajemanje podatkov
- s časovno periodo P se shranjujejo podatki o lokaciji (lat, lon), temperaturi in vlagi
- če je podatek o temperature izven podanega območja, se prikaže opozorilo v realnem času
- transport se ustavi s ponovnim branjem QR kode naročila na mestu dostave
- ob zaključku transporta se ustavi zajemanje podatkov; ti se pošljejo na strežnik podjetja
- analiza transporta (grafa temperature in vlažnosti, vsa obvestila,...)

Slika 2.1: Primer dobavnice, kateri smo v desni zgornji kot dodali QR kodo.[4]

Postopek nadzora smo definirali tako, da se začne pri dobavnici. To je dokument, ki vsebuje vse potrebne podatke o transportu - kdo je pošiljatelj, kdo prejemnik, vsebino tovora in drugo. Prav zaradi dejstva, da vsebuje vse potrebne informacije, lahko dobavnica služi kot osnova našega podatkovnega modela. Na sliki 2.1 lahko vidimo osnutek običajne dobavnice, z eno samo spremembo; dodali smo ji QR (angl. *Quick response*) kodo, s pomočjo katere bomo podatke o naročilu prikazali v mobilni aplikaciji. QR koda namreč hrani ID naročila, ki nam omogoča prenos ustreznega dokumenta s strežnika podjetja na mobilno napravo.

Na dobavnici je večina polj, ki jih bomo shranili v podatkovno bazo. Vseeno moramo poskrbeti še za nekaj dodatnih polj, ki jih dobavnica ne vsebuje.

[4] `https://www.papirnica-knjigarna.si/logistika-in-transport/2471-prevoznica-obrazec-410-3838884239493.html`

To so predvsem polja, povezana z zajemom meritev (lokacija, temperatura, vlažnost, obvestila,...) in bodo zato shranjena v ločeni entiteti.

Glavna entiteta podatkovnega modela je *Naročilo* (angl. *Order*), ki vsebuje naslednje atribute (imena atributov so zaradi enotnosti v angleškem jeziku):

- orderID - unikatni ID naročila
- title - opisni naslov naročila
- paid - ali je naročilo plačano
- sender - podatki o pošiljatelju
- receiver - podatki o prejemniku
- startLocation - x, y koordinate začetne lokacije
- endLocation - x, y koordinate končne lokacije
- vehicleTypeRequired - podatki o zahtevanem tipu vozila (navaden/s hladilnikom)
- cargo - seznam tovora
- cargoTempMax - zgornja meja priporočene temperature tovora
- cargoTempMin - spodnja meja priporočene temperature tovora
- date - datum kreiranja naročila
- status - status naročila (ni začeto, v teku, končano)
- text - polje za dodatne opombe
- transport - ID *Transport* dokumenta

Polje *transport* bo pri vsakem naročilu na začetku prazno. Šele ko bo naročilo končano, se bo v to polje shranila referenca na entiteto *Transport*. Tu so shranjeni vsi podatki, ki se zbirajo med transportom:

- idOrder - ID pripadajočega naročila
- measurements - zajete meritve
- alerts - seznam vseh obvestil
- startDate - čas ob začetku transporta
- endDate - čas ob koncu transporta
- duration - trajanje transporta
- vehicle - podatki o uporabljenem vozilu
- driverID - ID voznika
- delivered - ali je bilo naročilo uspešno dostavljeno
- text - opombe

2.3 Primer uporabe

Prevozno podjetje ima več voznikov, zato morajo biti vsi dodani v bazo uporabnikov. Na začetku se vsak voznik na podlagi uporabniškega imena vpiše v aplikacijo; to nam kasneje omogoči, da povežemo prevoznike z njihovimi opravljenimi transporti. Preden začne s transportom, mora uporabnik prebrati še QR kodo svojega vozila, da se podatki o vozilu (registrska številka, tip) shranijo v aplikacijo. Nato na prevzemni točki s pomočjo vgrajenega čitalnika prebere QR kodo z dobavnice blaga (Slika 2.1), pripravljenega za transport. Vsaka QR koda je unikatni ID naročila, na podlagi katerega mobilna aplikacija pridobi ustrezen dokument iz podatkovne baze prevozniškega

podjetja.

Uporabnik lahko prevzame več naročil hkrati - za vsako novo naročilo ponovno prebere njegovo QR kodo. Ko dobi podatke vseh naročil, lahko začne z dostavljanjem; takrat se tudi začne zbiranje meritev. Glede na določen časovni interval, mobilna naprava podatke o lokaciji prejema s pomočjo GPS senzorja naprave. Na podlagi teh koordinat, se nato izvede klic na spletno storitev *OpenWeatherMap*, ki vrne ustrezne podatke o temperaturi in vlagi za podano lokacijo.

Zajemanje meritev za dotičen prevoz se konča s ponovnim branjem QR kode naročila na ciljni lokaciji. Ko aplikacija ugotovi, da je prebran ID enak ID-ju enega izmed naročil *v izvajanju*, pošlje vse meritve, zbrane do tega trenutka, na strežnik podjetja. S tem se zajemanje meritev za to naročilo zaključi; če so *v izvajanju* še druga naročila, se zajemanje za ta še naprej nadaljuje (vključujoč vse zbrane podatke do tega trenutka). Celoten zajem se zaključi ob zadnji dostavi, ko se na strežnik se prenesejo podatki še zadnjega transporta. Ni več aktivnih naročil.

Med zajemom podatkov bo mobilna aplikacija opozarjala na morebitne nepravilnosti. Vsako naročilo ima določeno območje optimalnih razmer; podani sta minimalna in maksimalna temperatura. V primeru, da so v danem trenutku izmerjene vrednosti izven ustreznega območja, uporabnik v realnem času dobi obvestilo o odstopanju od priporočene temperature.

Zaradi narave zajetih podatkov - temperature in vlage iz ozračja - je uporabnost teh obvestil omejena. Opozorila so na primer smotrna pri uporabi navadnega kamiona (brez hladilnika) sredi poletja, ko lahko privzamemo, da sta temperaturi ozračja in notranjosti prostora za tovor podobni. Če je zgornja meja naročila 10 stopinj, zunanja temperatura ozračja pa 25 stopinj, potem je opozorilo uporabno. V primeru, da bi isto naročilo opravljali po-

zimi, opozoril ne bi bilo. Pri uporabi kamiona s hladilnikom zajete meritve niso uporabne, saj bi jih morali pridobivati s senzorja v notranjosti hladilnika.

Vse nepravilnosti so po končanem transportu vidne tudi pri analizi. Na voljo so grafi temperature in vlage in na podlagi lokacije v danem trenutku ima podjetje možnost preučiti, zakaj in kje je prišlo do nepravilnosti. Vsi zbrani podatki in njihova analiza lahko podjetju omogočijo optimizacijo procesov in posledično zmanjševanje stroškov in tveganj ter povečanje dobička.

Poglavje 3

Pregled tehnologij

Aplikacija bo namenjena mobilnim napravam z najbolj razširjenim mobilnim operacijskim sistemom Android. Razvita bo z uporabo Android Studia, uradnega Googlovega orodja za izdelavo aplikacij za Android. V nadaljevanju bomo opisali:

- operacijski sistem Android
- GPS (Global Positioning System)
- QR (Quick Response)
- spletne storitve (angl. Web services)

3.1 Operacijski sistem Android

Operacijski sistem Android je mobilni operacijski sistem, ki ga odlikujejo prilagodljivost, odprtokodnost in enostavnost za uporabo. Poganja več kot eno milijardo naprav po vsem svetu - od telefonov in tablic pa vse do pametnih ur, televizorjev, avtomobilov ter številnih drugih naprav.[1]

[1] `https://www.android.com/`

Sistem Android je produkt podjetja Google. Temelji na Linux jedru in je načrtovan za uporabo preko zaslonov na dotik, s pomočjo uporabnikovih akcij kot so potegi, pritiski in uporaba navidezne tipkovnice. Izvorna koda je objavljena pod odprtokodno licenco, kar je tudi eden glavnih razlogov za popularnost Androida.

Od leta 2007, ko je bil Android prvič predstavljen svetu, je bil operacijski sistem mnogokrat posodobljen. Vsaka posodobitev je povečala stabilnost programja, odpravila hrošče ali pa dodala nove funkcionalnosti. Vse večje izdaje so poleg številčne oznake prejele še kodno ime: ime slaščice v abecednem vrstnem redu glede na izdajo. V času pisanja diplomske naloge je zadnja različica *Android 8.0 Oreo*.

3.1.1 Zgodovina

Vse se je začelo z Googlovim nakupom podjetja Android, Inc. leta 2005 za najmanj 50 milijonov dolarjev.[2] Oktobra leta 2003 ustvarjeno podjetje se je sprva ukvarjalo z izdelavo naprednega operacijskega sistema za digitalne kamere, a kmalu ugotovilo, da je trg tovrstnih naprav premajhen in svojo pozornost preusmerilo v izdelavo sistema za pametne telefone, ki bi lahko konkuriral *Symbianu* in Microsoftovemu *Windows Mobile* sistemu.

Plod dela je bil najavljen leta 2007, skupaj z ustanovitvijo konzorcija *Open Handset Alliance* (OHA), namenjenega zagovarjanju in razvoju odprtih standardov za mobilne naprave. Konzorcij je povezal različna tehnološka podjetja, od proizvajalcev telefonov (HTC, Samsung, Sony), izdelovalcev elektronskih vezij (Qualcomm, Texas Instruments) pa do mnogih drugih (Google, Sprint, T-Mobile). Kot prvi produkt združenja je bil predstavljen prav operacijski sistem Android.[3]

[2] `http://www.nytimes.com/2015/05/28/technology/personaltech/a-murky-road-ahead-for-android-despite-market-dominance.html?_r=0`

[3] `http://www.openhandsetalliance.com/press_110507.html`

Zanimivo je, da so prvi prototipi zelo spominjali na telefone BlackBerry, saj niso imeli zaslona na dotik, temveč navigacijske smerne tipke in fizično QWERTY tipkovnico. Vse to se je spremenilo po predstavitvi Applovega prvega *iPhonea* v letu 2007, ki je nakazal nove smernice na področju pametnih telefonov in hitro je bila Androidu dodana podpora zaslonom na dotik.

Različice

Od leta 2007, ko je bil Android prvič predstavljen svetu, pa do danes, je šel sistem skozi mnogo iteracij. Vsaka izdaja je s seboj prinesla tudi nov API (*Application Programming Interface*) nivo, ki je razvijalcem zagotavljal, da bodo njihove mobilne aplikacije z uporabo najnovejših APIjev nemoteno in brez napak tekle na posodobljenem sistemu.

Vsaka nova različica je označena s številčno oznako, ki v večini primerov nakazuje, za kako velik skok naprej gre. Tako na primer različica 4.1 obeta le manjše spremembe ob nadgradnji z različice 4.0, medtem ko bi v primeru, da bi bila nadgradnja označena z oznako 5.0, pričakovali večje spremembe in nove funkcionalnosti. Vse od različice 1.5 naprej, imajo vse večje izdaje poleg številčne oznake tudi ime. Kodna imena si sledijo po abecednem vrstnem redu in so vedno izbrana iz družine slaščic. Prvi dve nekomercialni različici, Android 1.0 in 1.1, nista imeli določenih posebnih imen.

Velik problem vseh naprav z Androidom je, da po večini poganjajo starejšo različico. Google namreč vsako leto predstavi dve večji posodobitvi, proizvajalci telefonov pa zaradi različnih razlogov (prilagajanje uporabniškega vmesnika, varčevanje s sredstvi) svoje naprave posodobijo z večmesečno zamudo ali pa sploh ne. Tako ogromno naprav ostane brez varnostnih posodobitev, poleg tega pa uporabniki ne dobijo novih funkcionalnosti in izboljšav.

3.1.2 Mobilne aplikacije

Ena od pomembnejših lastnosti vsakega pametnejšega operacijskega sistema je nameščanje in poganjanje aplikacij. Za razliko od Applovega konkurenčnega sistema *iOS* (ta je omejen le na Applove naprave), ki je zunanje aplikacije predstavil šele v svoji drugi različici (iOS 2.0)[4], je Android - v skladu s svojim glavnim sporočilom odprte platforme, prilagajanja in kostumizacije - dovoljeval aplikacije zunanjih razvijalcev že od samega začetka.

Uporabniki najdejo in pridobijo aplikacije preko *Google Play* spletne trgovine, kjer so objavljene vse tiste, ki so bile odobrene s strani Googla. Lahko so brezplačne, plačljive in/ali vsebujejo nakupe znotraj aplikacije (angl. IAP, *in app purchases*). Uporabnik ima možnost namestitve aplikacij tudi iz ti. neznanih virov, vendar pa je to privzeto onemogočeno; potrebno je predhodno obkljukati izbiro v nastavitvah Androida.

Aplikacije so po večini napisane v *Javi*, lahko se uporabi tudi *C* in *C++* ali *Go* jezik, po novem pa tudi *Kotlin*. Od decembra 2014 naprej Google razvijalcem ponuja namensko orodje za izdelavo Android aplikacij, imenovano *Android Studio*. Pred tem je to delo opravljalo razvojno okolje *Eclipse* z vtičnikom ADT (*Android Development Tools*).

Android Studio je uradni IDE (*Integrated Development Environment*) Android platforme. Vsebuje git integracijo, kodne predloge, emulator s podporo mnogim napravam različnih velikosti in oblik ter še mnogo drugih pripomočkov, vse z enim samim ciljem: omogočiti razvijalcem čim enostavnejšo in čim hitrejšo pot od prvotne zamisli, pa do delujoče aplikacije v Google Play trgovini.

[4] `http://www.theverge.com/2011/12/13/2612736/ios-history-iphone-ipad`

3.2 GPS

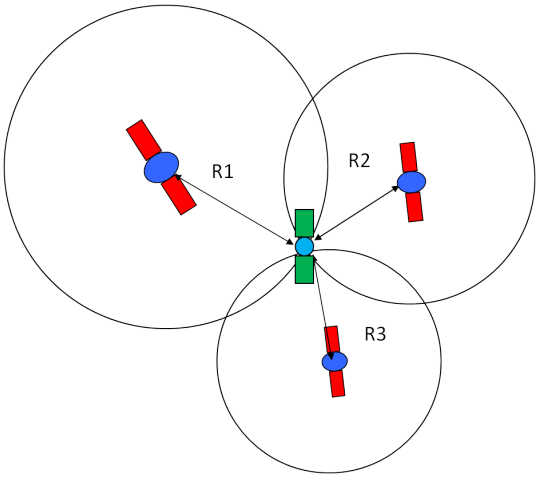
GPS (*Global Positioning System*) je globalni pozicijski sistem, ki omogoča vsem GPS sprejemnikom kjer koli na zemlji določiti njihovo točno geolokacijo s pomočjo signala namenskih satelitov, ki krožijo v zemljini orbiti. Tehnologijo so razvili za potrebe ameriške vojske, v 80-tih pa so jo odprli tudi za civilno uporabo. Polno funkcionalna je od leta 1993, ko je bilo v uporabi že 24 satelitov [2].

Trenutno je v vesolju 31 aktivnih satelitov, za določitev točne lokacije pa naprava z GPS sprejemnikom potrebuje signal vsaj štirih satelitov. V osnovi GPS sprejemnik namreč določi položaj z računanjem oddaljenosti od teh satelitov. Ti pošljejo signal, ki vsebuje točen čas, kdaj je bil signal odposlan. Ker sprejemnik hkrati dobi tudi podatek o natančni lokaciji satelitov ob pošiljanju, lahko na podlagi razlike med poslanim časom in pa časom prejema signala izračuna razdaljo do posameznega satelita. S pomočjo znane lokacije satelitov in pa razdalje do le-teh, lahko GPS natančno določi svoj položaj (do tri metre natančno) v treh dimenzijah (Slika 3.1) - x, y in z (nadmorska višina).

Ker se položaj določa z računanjem časa, ki ga porabi signal na svoji poti do sprejemnika, potrebujemo zelo natančne meritve časa: GPS oddajniki v ta namen uporabljajo atomske ure (izredno točne ure, ki za merjenje časa uporabljajo resonančne frekvence atomov in njenih rezonatorjev[5]), vendar to pri sprejemnikih ni izvedljivo. Tu nastopi signal s četrtega satelita, ki omogoči izračun točnega časa brez uporabe atomske ure. V primeru, da dobimo signal le s treh satelitov, bo sprejemnik še vedno lahko določil položaj, vendar pa ne bo tako natančen, saj se privzame, da smo na višini vodne gladine. Če je dejanska lokacija nekje v gorah, lahko to pomeni napako tudi več 100 metrov.[6]

[5] `http://www.tocnaura.si/atomska-ura`

[6] `https://www.maptoaster.com/maptoaster-topo-nz/articles/how-gps-works/`



Slika 3.1: Določanje lokacije s pomočjo treh satelitov.[7]

V želji po neodvisnosti od sistema, ki je pod polnim nadzorom ameriške vlade, so nastale konkurenčne rešitve, kot sta ruski sistem GLONASS (*Global Navigation Satellite System*) in projekt Evropske unije Galileo (GNSS, *Global Satellite Navigation System*). GPS sprejemniki v novejših napravah z uporabo tudi teh sistemov zagotavljajo večjo natančnost in hitrejšo prvo določitev lokacije.

3.3 Koda QR

Koda QR (*Quick Response*), je blagovna znamka 2-D matrične črtne kode (Slika 3.2), ki je bila sprva uporabljena v avtomobilski industriji na Japonskem. Leta 1994 jo je iznašel Denso Wave in zaradi možnosti hitrega branja podatkov, ki jih shranjuje, se je njena uporaba hitro razširila. Danes se uporablja tako v industriji, kot tudi v oglaševanju in drugih področjih za enostaven izpis.

how-gps-works.html

[7] https://www.nasa.gov/sites/default/files/gps_signals.png

Slika 3.2: Primer QR kode, ki v zapisu skriva število 42.

Zaradi pametnih mobilnih naprav, ki lahko z uporabo kamere in ustrezne programske opreme v trenutku razpoznajo vsako QR kodo, so le-te izredno priročne. Namesto, da se na oglaševalski plakat zapiše dolg spletni naslov do izdelka, ki ga bo uporabnik vpisoval 5 minut, se raje zraven izdelka nalepi QR kodo, ki jo uporabnik s telefonom pridobi v trenutku ter nato prikaže spletno stran. Prav preprosta pridobitev informacije iz QR kode je razlog, da jo bomo vključili v našo rešitev.

QR koda je sestavljena iz črnih kvadratov različnih velikosti na belem ozadju. S pomočjo kamere ali čitalca in uporabe *Reed-Solomonove odprave napak* se iz QR kode izlušči podatke, ki so zapisani tako v vertikalnih kot tudi horizontalnih komponentah slike.

3.4 Spletne storitve

Spletna storitev (angl. *Web Service - WS*), je storitev, ki omogoča komunikacijo med napravami v omrežju. Gre za programski sistem, ki je prilagojen za komunikacijo med napravami, z uporabo standariziranih formatov, ki jih razumejo, npr. XML in JSON [7].

Čeprav so načeloma glavne tehnologije, ki sestavljajo spletno storitev, XML (format podatkov), SOAP (prenos podatkov), WSDL (opis storitev, ki so na voljo) in UDDI (seznam storitev, ki so na voljo), pa obstajajo tudi enostavnejše spletne storitve, ki teh protokolov ne vključujejo. Eno od teh smo uporabili pri našem produktu, zato si jo bomo nekoliko podrobneje pogledali.

3.4.1 OpenWeatherMap

OpenWeatherMap je spletna storitev, ki prek enostavnega APIja nudi podatke o vremenu. Na voljo so podatki o trenutnem vremenu, napovedih in preteklem vremenu za več kot 200,000 mest ali katero koli lokacijo na svetu. Podatke, pridobljene iz različnih virov, lahko neodvisno preverijo s surovimi podatki z več kot 40,000 vremenskih postaj po celem svetu.[8]

Podjetje poleg plačljivih paketov za manjše projekte ponuja tudi brezplačno zbirko, ki omogoča do 60 klicev API-ja na minuto in nudi vse podatke o trenutnem vremenu, vremensko napoved in še mnogo več. Za odgovor lahko izbiramo med formati JSON, XML ali HTML. Dokumentacija je odlična, tako da je uporaba API klicev zelo enostavna.

Prav preprosta uporaba, odlična dokumentacija in brezplačen paket z vsemi vključenimi meritvami so nas prepričali, da smo to spletno storitev uporabili v naši aplikaciji.

[8] `http://openweathermap.org/about`

Poglavje 4

Razvoj

Razvoj končnega produkta je sestavljen iz dveh delov - mobilne aplikacije in strežnika. Poudarek je na mobilni aplikaciji, saj je to glavna tema diplomskega dela. Vseeno je bilo potrebno postaviti tudi strežnik, ki mobilni aplikaciji prek REST (angl. *representational state transfer*) klicev omogoča komunikacijo s podatkovno bazo.

Pri razvoju smo uporabljali git - sistem za nadzor verzij (angl. *VCS, version control system*). Čeprav je na projektu delala le ena oseba, je git prišel zelo prav pri menjavi naprav, saj se je delo lahko začelo na eni napravi, nato pa brez težav nadaljevalo na drugi. Hkrati je služilo tudi kot varovalo v primeru izgube lokalnih podatkov.

Celoten razvoj je potekal v Linux okolju. Za razvoj mobilne aplikacije smo uporabili programski jezik Java in Googlov uradni IDE za razvoj android aplikacij, Android Studio. Testiranje je potekalo na pametnem telefonu *LG G4*, na katerem je tekla uradna različica Android 6.0 Marshmallow. Telefon je bil izdan aprila 2015 in ima zaslon resolucije 2560x1440, na katerega stvari izrisuje čipovje Snapdragon 808 s 3 GB delovnega pomnilnika (RAM).[1] Upoštevaje dejstvo, da telefon ni najnovejši, gre po našem mnenju

[1] `http://www.gsmarena.com/lg_g4-6901.php`

za solidno testno napravo, ki lahko v grobem predstavlja sredino ogromnega trga mobilnih naprav s sistemom Android.

Strežnik smo postavili s pomočjo platforme *Heroku*. Gre za oblačno storitev *PaaS* (angl. *Platform as a service*), ki razvijalcem omogoča hitro in skalabilno postavitev aplikacij. Za gostovanje naše podatkovne baze smo uporabili *mLab*. Ta storitev omogoča brezplačno gostovanje *MongoDB* podatkovnih baz vse do velikosti 500 MB.

Strežnik izpostavlja REST API. Kaj to pomeni? Predstavitveni prenos stanja (angl. *Representational state transfer - REST ali RESTful*) je način sporazumevanja med računalniškimi napravami na spletu. Gre za arhitekturni stil, ki spodbuja k uporabi protokola HTTP in njegovih zahtevkov:

- GET - dobi vnos
- POST - ustvari nov vnos
- PUT - posodobi vnos
- DELETE - izbriši vnos

Tako v našem primeru GET API klic za naročilo z ID-jem 20 izgleda takole:

`https://diploma-server-rest.herokuapp.com/api/orders/20`.

Nazaj dobimo odgovor v JSON formatu (Slika 4.1). Z uporabo standardnih operacij in protokola brez stanja nam RESTful spletna storitev zagotavlja hitro in zanesljivo delovanje.

```
{
  "_id": "5974bf91734d1d6202a92447",
  "orderID": 20,
  "title": "Narocilo st. 20",
  "sender": {
    "customerID": 2,
    "name": "Kmetija Pr' Čib",
    "address": {
      "street": "Ulica Eve Lovše 1",
      "city": "Maribor",
      "cityCode": 2000,
      "country": "Slovenia",
      "_id": "59b610cdbe1c10001264ecd1"
    },
    "_id": "59b610cdbe1c10001264ecd0"
  },
  "receiver": {
    "customerID": 1,
    "name": "Hofer",
    "address": {
      "street": "Mesarska cesta 10",
      "city": "Ljubljana",
      "cityCode": 1000,
      "country": "Slovenia"
    }
  },
  "startLocation": {
    "x": 46.0569,
    "y": 14.5058,
    "_id": "59b610cdbe1c10001264eccf"
  },
  "endLocation": {
    "x": 46.5547,
    "y": 15.6459,
    "_id": "59b610cdbe1c10001264ecce"
  },
  "cargo": {
    "minTemp": 5,
    "maxTemp": 13,
    "_id": "59b610cdbe1c10001264eccd",
    "items": []
  },
  "date": "2017-07-12T12:00:00.196Z",
  "text": "Previdno ravnanje z blagom!",
  "transport": "597c62ed734d1d58c9ab91e7",
  "status": 0,
  "transportID": null,
  "vehicleTypeRequired": 0,
}
```


Slika 4.1: Primer JSON odgovora strežnika na API klic (na sliki entiteta *Order*).

4.1 Načrtovanje mobilne aplikacije

Preden začnemo z izdelavo aplikacije za Android, se moramo najprej odločiti, katere različice operacijskega sistema bomo podprli. Pri predstavitvi operacijskega sistema Android smo spoznali, da veliko naprav ne dobi najnovejših nadgradenj. Kaj to pomeni za razvijalce mobilnih aplikacij? Skupaj z nadgradnjami za naprave, se ob novih različicah dodajo tudi novi APIji za razvijalce, ki omogočajo nove funkcionalnosti. Ti se morajo tako odločiti, kaj je še zadnji API nivo, ki ga bodo podprli; starejša kot je različica, manj novejših APIjev bo podpirala.

Na srečo podjetje Google ponuja rešitev z uporabo knjižnice *Support Library*. Ta namesto razvijalcev poskrbi za združljivost novejših APIjev s starejšimi napravami [6]. Vse, kar mora razvijalec narediti, je definirati *minSdkVersion* (minimalni API nivo) in *targetSdkVersion* (ciljani API nivo). MinSdkVersion je minimalni API nivo, ki ga aplikacija še podpira - če ima neka naprava različico sistema, katere API nivo je nižji, aplikacije ne bo mogoče namestiti. TargetSdkVersion napravi pove, da naj ne omogoči nobenih združljivostnih knjižnic, kadar je podan API nivo enak kot na napravi. Pri aplikaciji smo minimalni API nivo postavili na 15, kar sovpada z Android verzijo 4.0.3. S tem zajamemo kompatibilnost z več kot 99% napravami na trgu [1]. Ciljni API smo nastavili na 25 (ob začetku razvoja najnovejša različica, *Android 7.0 Nougat*).

Posebno pozornost smo namenili uporabniški izkušnji (angl. *User Experience - UX*). Želimo, da aplikacija deluje hitro, zanesljivo in je prijetna na pogled in enostavna za uporabo. Velikokrat se zgodi, da se pri mobilnih aplikacijah, ki so namenjene resnejšemu delu, da prednost funkciji pred izgledom. Posledično so aplikacije zelo zmogljive, vendar pa neintuitivne in zapletene za uporabo, z neokusnimi uporabniškimi vmesniki.

Naša rešitev mora biti oboje - funkcionalna in lepa. Zato smo se odločili

za uporabo *Material Designa*, Googlove ideje, kako naj izgleda in deluje uporabniški vmesnik. V uporabi je privzeto od različice *Android 5.0 Lollipop*, ker uporabnikom omogoča poenoteno izkušnjo pri uporabi Android naprav in njihovih aplikacij, z elementi, ki jih uporabniki poznajo in vedo, kakšne so njihove funkcije.

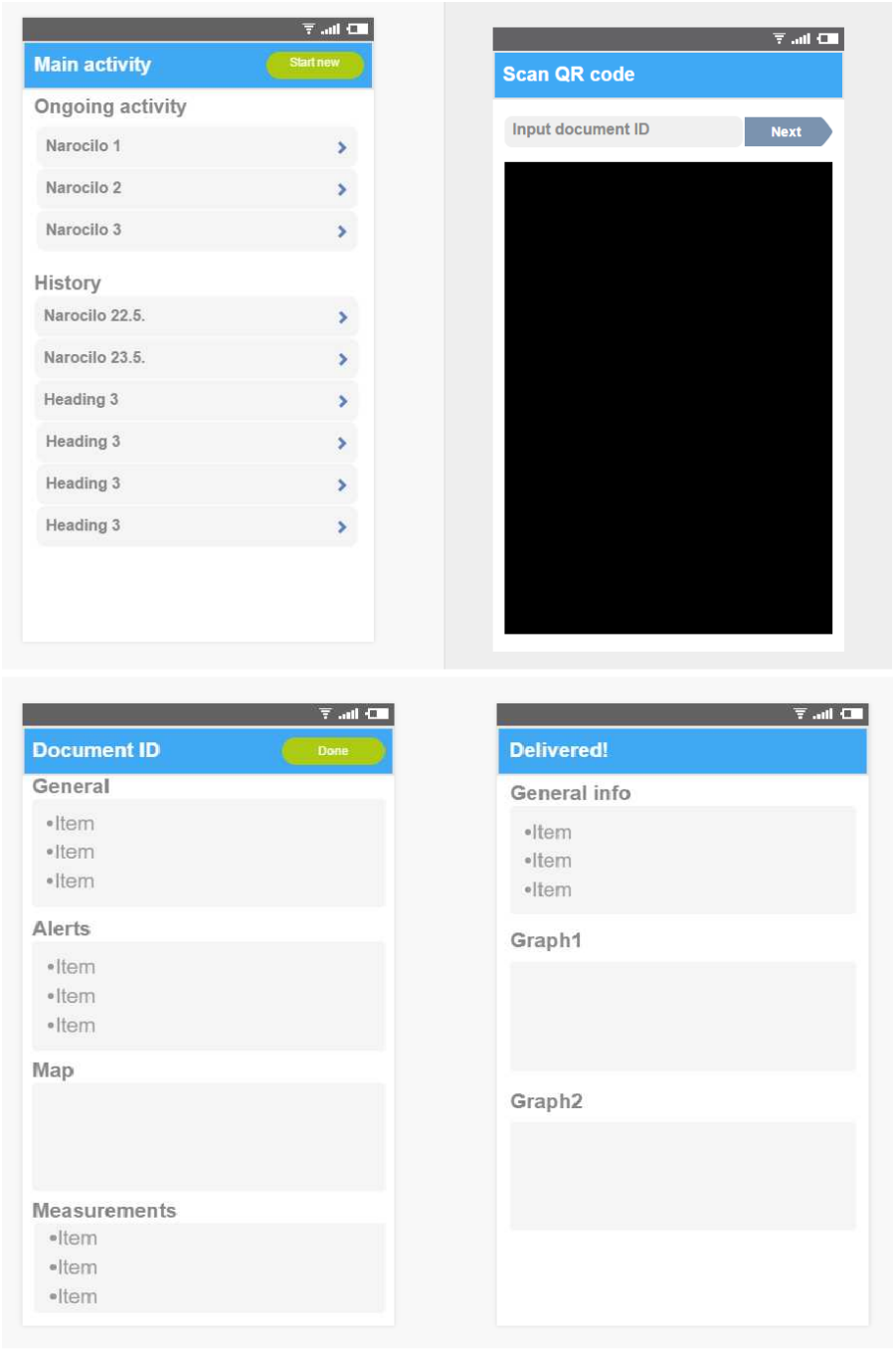
Glede na zahtevane funkcionalnosti, smo na začetku določili štiri glavne zaslone aplikacije:

- vsa naročila (angl. *Main activity*) - prikazuje vsa naročila v teku ter že opravljene transporte
- branje QR kode (angl. *Scan QR code*) - omogoča branje ali ročni vpis ID-ja, na podlagi katerega prejmemo ustrezni dokument s strežnika podjetja
- podrobnosti naročila (angl. *Document ID*) - prikaz vseh podrobnosti naročila
- podrobnosti transporta (angl. *Delivered!*) - prikaz vseh podrobnosti po koncu transporta

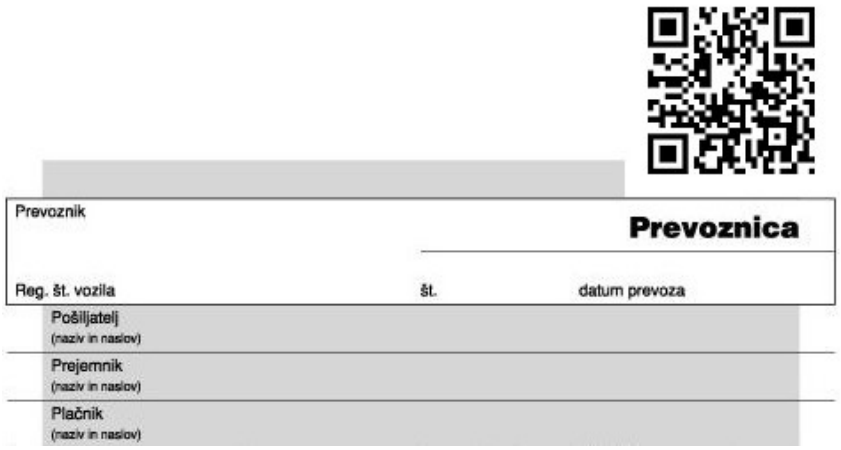
To so najpomembnejši deli aplikacije, zato smo izdelali grobe osnutke teh zaslonov (angl. *wireframe*), kot so prikazani na sliki 4.2). Med izdelavo aplikacije se je pokazala potreba še po enem zaslonu za nadzor transporta - prikaz statusa zajemanja meritev, njihove vrednosti, koliko časa transport že traja, katera naročila so aktivna,...

4.2 Implementacija

Pri pregledu implementacije načrtovanih funkcionalnosti se bomo osredotočili na tri glavne korake v aplikaciji, ki so zanimivi z vidika uporabljenih tehnologij: branje QR kod, zajemanje meritev, pridobivanje lokacije naprave in



Slika 4.2: Grobi osnutki glavnih zaslonov aplikacije (zgoraj od leve: vsa naročila, branje QR kode; spodaj od leve: podrobnosti naročila, podrobnosti transporta).



Slika 4.3: Primer QR kode na dobavnici.

klicanje zunanje spletne storitve.

4.2.1 Branje QR kode

Prevozno podjetje ima več voznikov, zato morajo biti vsi dodani v bazo uporabnikov. Na začetku se vsak voznik na podlagi uporabniškega imena vpiše v aplikacijo; to nam bo kasneje omogočilo, da povežemo prevoznike z njihovimi transporti. Nato na prevzemni točki s pomočjo kamere mobilne naprave prebere QR kodo z dobavnice tovora, pripravljenega na transport.

Vsaka QR koda na dobavnici (Slika 4.3) je unikatni ID, na podlagi katerega mobilna aplikacija pridobi ustrezen dokument (naročilo) iz podatkovne baze prevozniškega podjetja. Entiteta Order (slo. *naročilo*) namreč vsebuje polje *orderID*, ki se mora ujemati s podatki, zakodiranimi v QR kodi na dobavnici.

Kako pa lahko mobilna naprava prepozna vsebino QR kode? Obstaja elegantna rešitev: uporaba *Mobile Vision APIja*, ogrodja, ki omogoča enostavno implementacijo zaznave obrazov, črtnih kod, besedila,...[3] Gre za ogrodje, ki

je na voljo vsem razvijalcem, ki v svojo aplikacijo vključijo storitev *Google Play Services* - skupek z Googlove strani pripravljenih APIjev, ki med drugim vključujejo *Google Maps* in *Google+* integracijo [4].

Poglejmo si konkreten primer naše implementacije. Najprej uvozimo razred, ki nas zanima (datoteka *ReadQRActivity.java*):

```
import com.google.android.gms.vision.barcode.BarcodeDetector;
```

Nato inicializiramo detektor...

```
BarcodeDetector barcodeDetector =
new BarcodeDetector.Builder(this)
     .setBarcodeFormats(Barcode.QR_CODE)
        .build();
```

... in čakamo na detekcije:

```
@Override
public void receiveDetections(Detections<Barcode> detections) {
    final SparseArray<Barcode> barcodes =
        detections.getDetectedItems();
   if (barcodes.size() != 0 && first == 0) {
        first = 1;
        Intent intent = new Intent(
            getBaseContext(),
            OrderItemActivity.class
        );
        intent.putExtra("isData", false);
        intent.putExtra("id", barcodes.valueAt(0).displayValue);
        startActivity(intent);
    }
}
```

Iz zgornje kode lahko razberemo, da takoj, ko zaznamo detekcijo, prenehamo z branjem QR kode in zaženemo nov *activity*, kateremu kot parameter podamo vrednost, ki jo je vsebovala prebrana QR koda. S to vrednostjo nato oblikujemo zahtevek na REST strežnik, ta pa iz podatkovne baze vrne ustrezno naročilo. Če zahtevanega naročila ni mogoče najti (ker npr. ne obstaja), se v aplikaciji prikaže obvestilo o napaki.

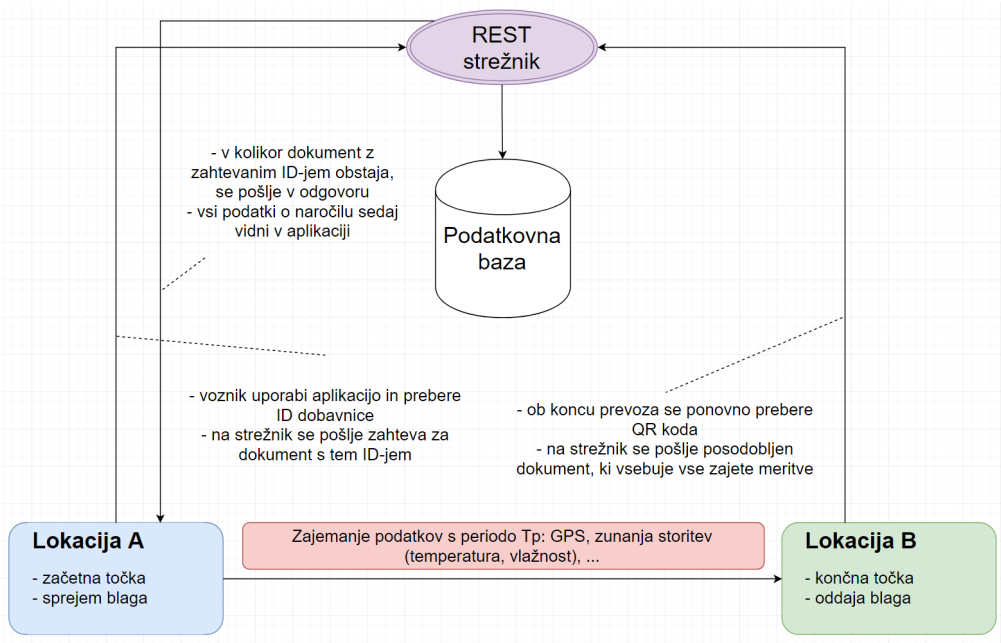
V primeru, ko QR kode ni mogoče prebrati (preslaba svetloba, pokvarjena kamera na mobilni napravi, itd.), je dodana tudi možnost ročnega vnosa ID-ja dokumenta.

4.2.2 Zajemanje meritev

Ko imamo naročilo preneseno na mobilno napravo, lahko začnemo z zajemanjem meritev. Glavni cilj zajemanja je, da se ves čas, ko je aktivno eno ali več naročil, zbira podatke o lokaciji naprave, zunanji temperaturi in vlagi v zraku. Da bi se to lahko nadaljevalo tudi po tem, ko uporabnik ugasne zaslon naprave ali zapusti aplikacijo, smo za zajemanje meritev uporabili proces, ki teče v ozadju (*IntentService*).

Zajemanje poteka skozi celoten čas transporta: če imamo le eno naročilo, bo torej potekalo od začetka tega naročila na lokaciji A, pa do zaključka naročila na lokaciji B. Celoten potek transporta je prikazan na sliki 4.4.

Če imamo več naročil, ki potekajo vzporedno, je potem v danem trenutku lahko aktivnih tudi več zajemanj? Ne; rešitev smo zasnovali tako, da za vsa zajemanja uporabimo le en proces v ozadju, ki vsako novo meritev doda v seznam. Ob začetku transporta si za posamezno naročilo le zabeležimo, pri kateri meritvi v seznamu se je transport pričel, ob koncu pa shranimo še indeks zadnje meritve. Tako je seznam meritev za nek transport kar podseznam vseh meritev, omejen z začetnim in končnim indeksom zajemanja

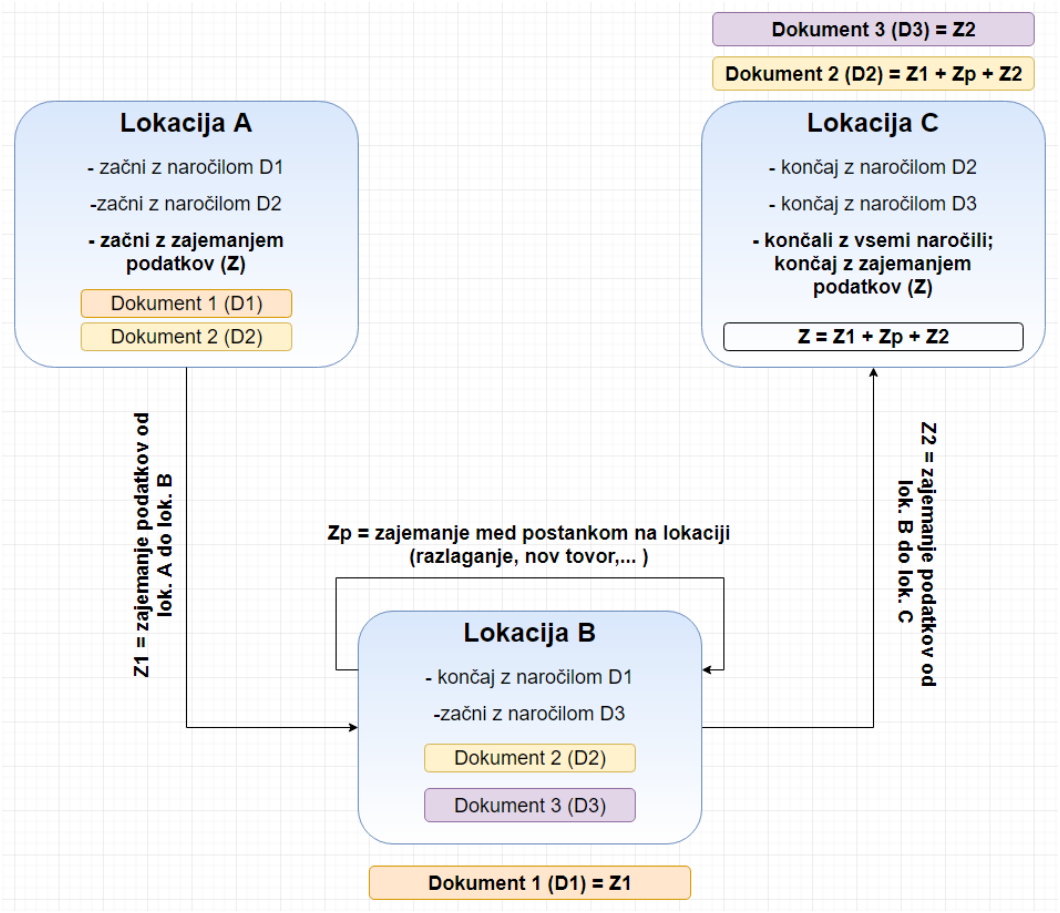


Slika 4.4: Transport enega naročila med dvema lokacijama.

(datoteka *MonitorService.java*):

```
public static JSONArray
    getMeasurements(int startIndex, int endIndex) {
        JSONArray subArray = new JSONArray();
        for (int i = startIndex; i < endIndex; i ++) {
            try {
                subArray.put(dataJSON.get(i));
            } catch (JSONException e) {
                e.printStackTrace();
            }
        }
        return subArray;
    }
```

Poglejmo si proces zajemanja meritev za več naročil naenkrat na sliki 4.5. Na lokaciji A - mestu prevzema naročila - uporabnik prebere kodi in prenese



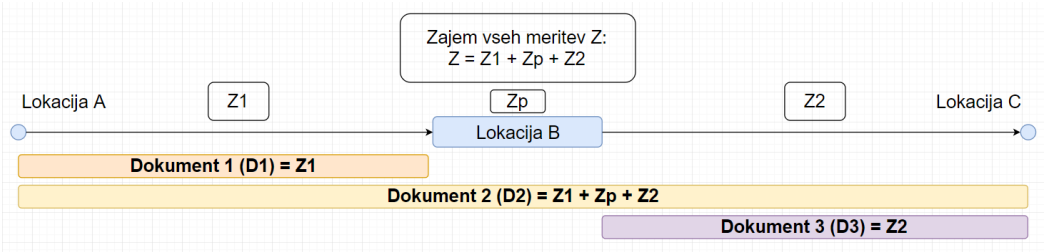
Slika 4.5: Zajem meritev pri več naročilih.

dokumenta D1 in D2. S pritiskom na gumb v aplikaciji naznani, da je bil tovor uspešno naložen in da se začenja faza transporta; začne se zajemanje podatkov.

Vendar pa dokumenta D1 in D2 nimata iste končne lokacije - predpostavimo, da je potrebno D1 dostaviti na lokacijo B, D2 pa na lokacijo C. Ob prihodu na lokacijo B, se ponovno prebere ID koda dokumenta D1 in zbiranje podatkov za to naročilo se zaključi - kreira se nov JSON objekt *transport*, ki se pošlje na strežnik z vsemi podatki o končanem transportu (datoteka *MonitorService.java*):

```
private void saveMeasurements() {

        ...

        for(OrderDocumentJSON ord : orders) {
            int startIndex = ord.getStartIndex();
            ord.setEndIndex(endIndex);
            JSONArray measurements =
                getMeasurements(startIndex, endIndex);

            ...

            // create new object transport
            JSONObject tr = addNewTransportObject(
                ord.getId(),
                measurements,
                alertMsgs,
                startTime,
                endTime,
                delivered
            );

            ...

        }
    }
```

Meritve za dokument D2 se še naprej zajemajo ves ta čas, tudi med postankom, ki je potreben za raztovarjanje tovora naročila D1. Poglejmo primer, kjer se na lokaciji B prevzame še novo naročilo D3. Spet se ob branju ID kode in prenosu dokumenta začne zajem podatkov za naročilo D3. To se nadaljuje vse do prihoda na lokacijo C, kjer se zaključita obe naročili, D2



Slika 4.6: Časovnica zajema meritev posameznih transportov.

in D3. S tem se zajem podatkov zaustavi.

Vendar pa naročili nimata enakega časovnega okvirja - D3 vsebuje meritve od lokacije B do lokacije C, D2 pa se posodobi z zbranimi podatki celotnega transporta, saj se je naročilo začelo na lokaciji A in zaključilo na lokaciji C. Zajete meritve dokumenta D2 tako vsebujejo:

- meritve od lokacije A do lokacije B
- meritve, zajete med postankom na lokaciji B (čeprav gre za postanek, je tovor naročila D2 še vedno naložen na prevoznem vozilu in nas zajete meritve zato še vedno zanimajo)
- meritve od lokacije B do lokacije C

Končno časovnico zajema meritev za vse tri transporte lahko vidimo na sliki 4.6.

4.2.3 Določanje položaja in pridobivanje podatkov iz zunanje spletne storitve

Da bi dobili točne podatke o vremenu iz zunanje storitve, najprej potrebujemo lokacijo naprave. Tako kot pri branju črtnih kod, smo si tudi tukaj pomagali s storitvijo *Google Play Services* in sicer smo uporabili *FusedLocationProviderApi*. Ta nam omogoča, da se "naročimo" na posodobitve lokacije, z določenim časovnim intervalom (datoteka *MonitorService.java*):

```
// get continous location updates
LocationServices.FusedLocationApi
    .requestLocationUpdates(
        mGoogleApiClient,
        mLocationRequest,
        this
    );
```

V odgovoru dobimo x in y koordinate lokacije naprave. Ta je določena s pomočjo senzorjev GPS in brezžičnih omrežij (Wi-Fi). Če ima uporabnik izklopljen GPS ob začetku zajemanja meritev, se prikaže opozorilo, ki poziva k vklopu sprejemnika GPS. V nasprotnem primeru se izračuna približek lokacije na podlagi Wi-Fi omrežij v bližini. Tako lahko vsakič, ko prejmemo novo posodobitev lokacije, iz odgovora določimo GPS koordinate in kličemo zunanjo storitev (datoteka *MonitorService.java*):

```
@Override
public void onLocationChanged(Location location) {
    mLastLocation = location;
    mLastUpdateTime = new Date();
    // get coordinates
    lastLat = Double.parseDouble(
        String.valueOf(location.getLatitude())
    );
    lastLon = Double.parseDouble(
        String.valueOf(location.getLongitude())
    );
    getWeatherData(lastLat, lastLon, mLastUpdateTime);
}
```

Za pridobivanje podatkov o ozračju uporabimo brezplačno storitev *OpenWeatherMap* (poglavje 3.4.1). Ta na podlagi vnesenih GPS koordinat vrne vremenske podatke za izbrano območje, kot je prikazano na sliki 4.7. Nas

```
{
  "coord": {
    "lon": 139,
    "lat": 35
  },
  "sys": {
    "country": "JP",
    "sunrise": 1369769524,
    "sunset": 1369821049
  },
  "weather": [
    {
      "id": 804,
      "main": "clouds",
      "description": "overcast clouds",
      "icon": "04n"
    }
  ],
  "main": {
    "temp": 289.5,
    "humidity": 89,
    "pressure": 1013,
    "temp_min": 287.04,
    "temp_max": 292.04
  },
  "wind": {
    "speed": 7.31,
    "deg": 187.002
  },
  "rain": {
    "3h": 0
  },
  "clouds": {
    "all": 92
  },
  "dt": 1369824698,
  "id": 1851632,
  "name": "Shuzenji",
  "cod": 200
}
```


Slika 4.7: Primer JSON odgovora spletne storitve na API klic.

zanimajo le polja *temp, humidity* in *pressure* (tega sicer nikjer ne uporabimo in prikažemo, a morda nekoč to omogočimo s posodobitvijo).

Kot je razvidno iz spodaj podane kode, najprej sestavimo URL za API klic (dodamo GPS koordinate, naš ID za uporabo APIja in izberemo merski sistem) in nato pošljemo zahtevek. Iz dobljenega odgovora preberemo polja, ki nas zanimajo in jih dodamo v seznam meritev (datoteka *MonitorService.java*):

```
// build url
String url ="http://api.openweathermap.org/data/2.5/weather?"
    + "lat=" + lat
    + "&lon=" + lng
    + "&APPID=" + weatherAppID
    + "&units=metric";
```

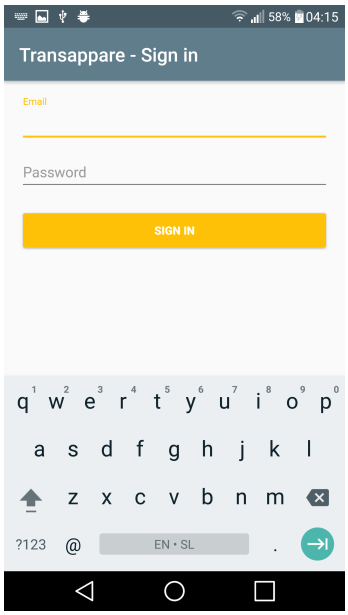
```
// Request a string response from the provided URL.
StringRequest stringRequest = new StringRequest(
    Request.Method.GET, url, new Response.Listener<String>() {
         @Override
         public void onResponse(String response) {
             Log.v(TAG, "Weather response is: "+ response);
             mLastWeatherUpdate = response;

             // get relevant data from weather report
             JSONObject weatherObject;
             try {
                 weatherObject = new JSONObject(response);

                 lastTemp = weatherObject
                     .getJSONObject("main")
                     .getDouble("temp");

                 lastHumidity = weatherObject
                     .getJSONObject("main")
                     .getDouble("humidity");

                 lastPressure = weatherObject
                     .getJSONObject("main")
                     .getDouble("pressure");

             } catch (JSONException e) {
                 e.printStackTrace();
             }

             // once we get weather data, add it to JSON
```

```
            addDataToJSON(
                lastTemp,
                lastHumidity,
                lastPressure,
                lat,
                lng,
                updateTime
            );

            ...

        }
    }
```

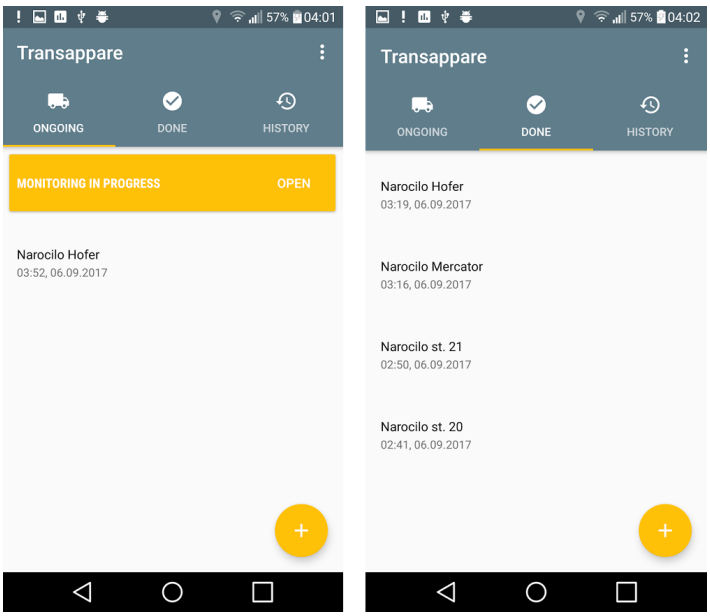
4.3 Testiranje

Poglejmo si, kako izgleda uporaba aplikacije *Transappare* v praksi. Pri prvi uporabi mora uporabnik vnesti s strani podjetja dodeljen elektronski naslov in pripadajoče geslo. Na podlagi te kombinacije se bo nato lahko vpisal v aplikacijo in vsi opravljeni transporti bodo označeni z uporabnikovim ID-jem (Slika 4.8). Vpis je potreben le pri prvi uporabi, saj si aplikacija zabeleži uporabnikove podatke.

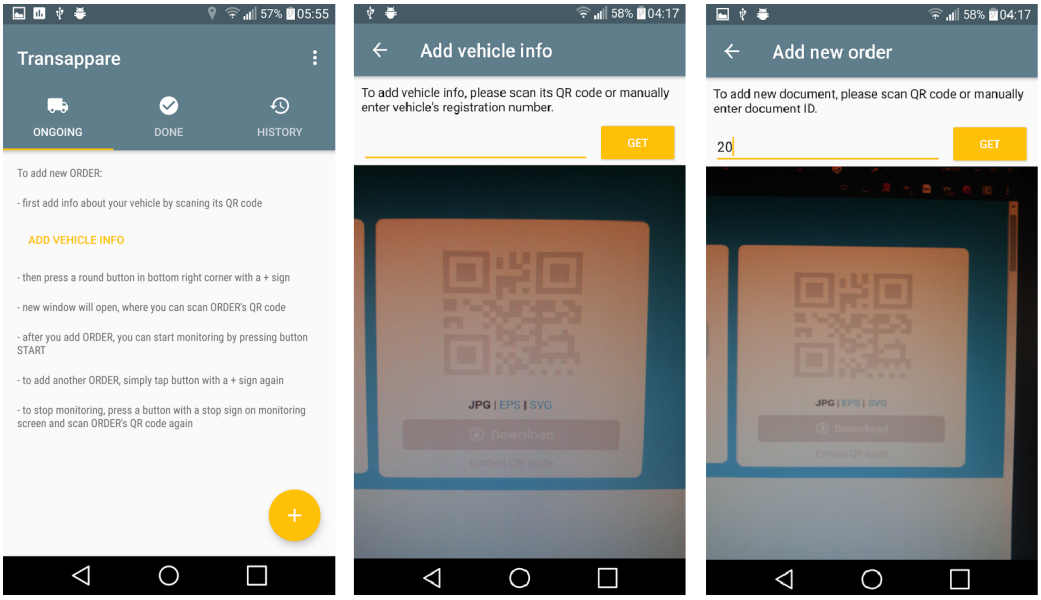
Tako nas ob ponovni uporabi vedno najprej pozdravi začetni zaslon, ki je tudi glavno okno aplikacije. Gre za zaslon, kjer lahko vidimo vsa naročila: od tistih, ki so v izvajanju - zavihek *Ongoing*, pa vse do tistih, ki so bila zaključena že nekaj časa nazaj - zavihek *Done*. Uporabnik lahko z vodoravnimi potegi prsta menja med tremi zavihki - *Ongoing* ali *V izvajanju* (vsa aktivna naročila), *Done* ali *Končano* (vsa nedavno zaključena naročila) ter *History* ali *Zgodovina* (ostala, starejša končana naročila), kot je prikazano na sliki 4.9.



Slika 4.8: Uporabnik se z elektronskim naslovom prijavi v aplikacijo.



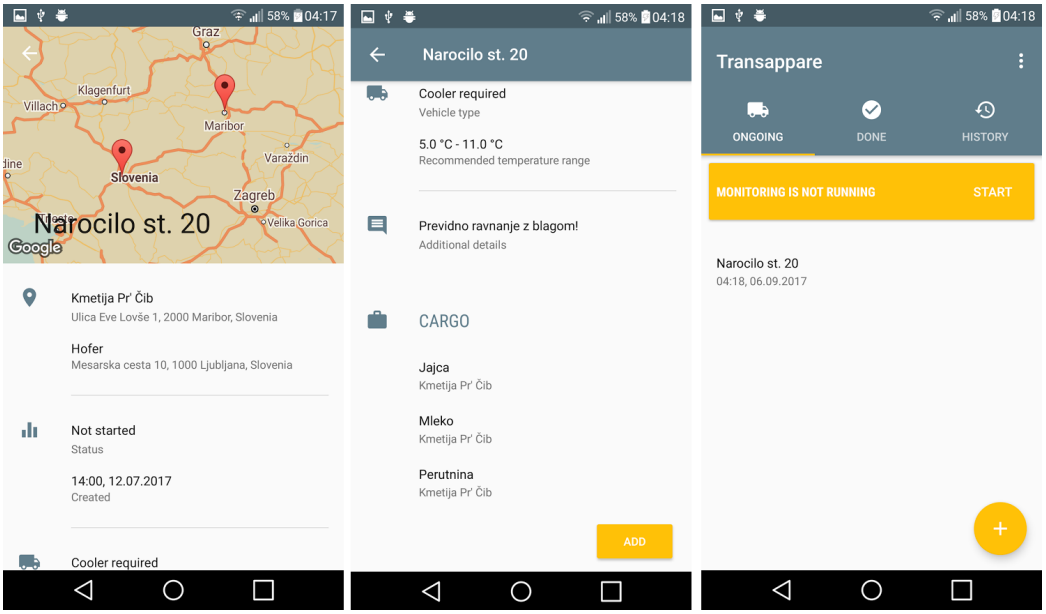
Slika 4.9: Na sliki levo zavihek *Ongoing* z aktivnimi naročili, na desni pa zavihek *Done* in končana naročila.



Slika 4.10: Če ni aktivnih nobenih naročil, uporabniku prikažemo pomoč (levo); sledi dodajanje vozila (na sredi) in dodajanje naročila (desno).

V primeru, da ni nobenega aktivnega naročila, nas v prvem zavihku pozdravi besedilo, ki na kratko opisuje, kako poteka uporaba aplikacije, hkrati pa vsebuje tudi gumb, ki uporabniku omogoča, da vnese informacije o svojem vozilu, bodisi z branjem njegove QR kode bodisi z ročnim vnosom registrske številke vozila. Vnašanje teh podatkov je pogoj, da lahko uporabnik v naslednjem koraku doda naročilo (Slika 4.10).

Dodajanje naročila poteka podobno kot dodajanje podatkov o vozilu; uporabnik ob prevzemu tovora z dobavnice prebere QR kodo ali ročno vnese ID naročila, ta pa se s strežnika podjetja prenese na mobilno napravo. Uporabnik lahko nato pregleda vse podatke, ki so na voljo, in s pritiskom na tipko *Add* ali *Dodaj* potrdi, da gre za pravi dokument, ta pa se nato shrani v podatkovno bazo mobilne naprave (Slika 4.11). V primeru, da se doda le eno naročilo, lahko sedaj uporabnik pritisne na gumb *Start* in zajemanje meritev se bo pričelo (Slika 4.11, desno). Odpre se novo okno, kjer so v realnem času na voljo vse relevantne informacije o transportu: trajanje zajema

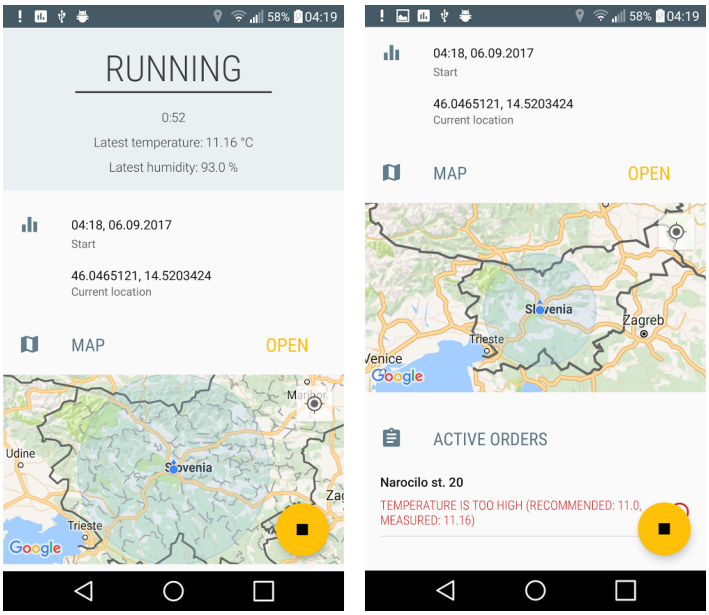


Slika 4.11: Pregledamo naročilo in ga dodamo v bazo naprave. S pritiskom na gumb *START* se prične zajemanje meritev (desno).

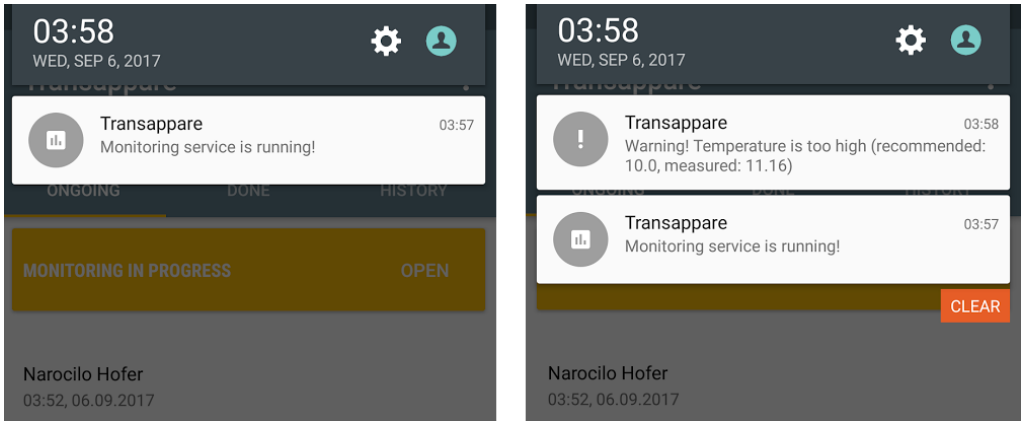
podatkov, zadnje vrednosti meritev temperature in vlage, zemljevid z našo lokacijo, stanje vseh aktivnih naročil in drugo (Slika 4.12).

Če je pri zajemanju meritev vrednost temperature izven podanega območja, potem se uporabniku prikaže sistemsko obvestilo, ki opozarja na odstopanja. Klik na obvestilo prikaže uporabniku okno zajemanja meritev, kjer lahko takoj vidi, pri katerem naročilu je prišlo do težav (Slika 4.13).

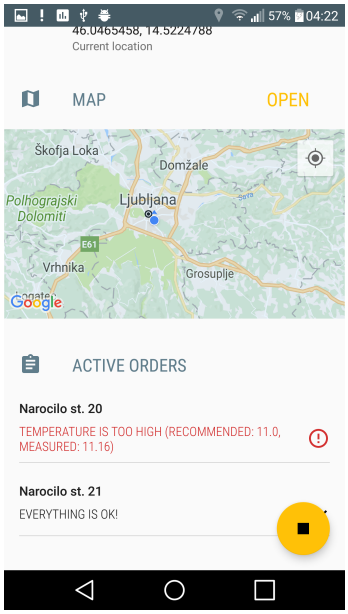
Med transportom se lahko kadarkoli doda novo naročilo, ki se pridruži ostalim aktivnim transportom (Slika 4.14). Prav tako se lahko naročilo tudi kadar koli zaključi, neodvisno od drugih. Ko pridemo na cilj in želimo zaključiti z zajemanjem meritev, moramo še enkrat prebrati QR kodo z dobavnice naročila (ali pa ID naročila vpisati ročno). Ko to opravimo, se zajemanje zaključi, vsi podatki se pošljejo na strežnik, nam pa je nemudoma na voljo analiza transporta, kjer vidimo grafa temperature in vlage, statistiko in ze-



Slika 4.12: Levo: spremljanje zajemanja meritev (zemljevid lahko po želji približamo). Desno: izmerjena temperatura je izven priporočenega območja aktivnega naročila.



Slika 4.13: Levo: obvestilo je vidno vse dokler se v ozadju zajemajo meritve. Desno: če temperatura ni ustrezna, se prikaže obvestilo.

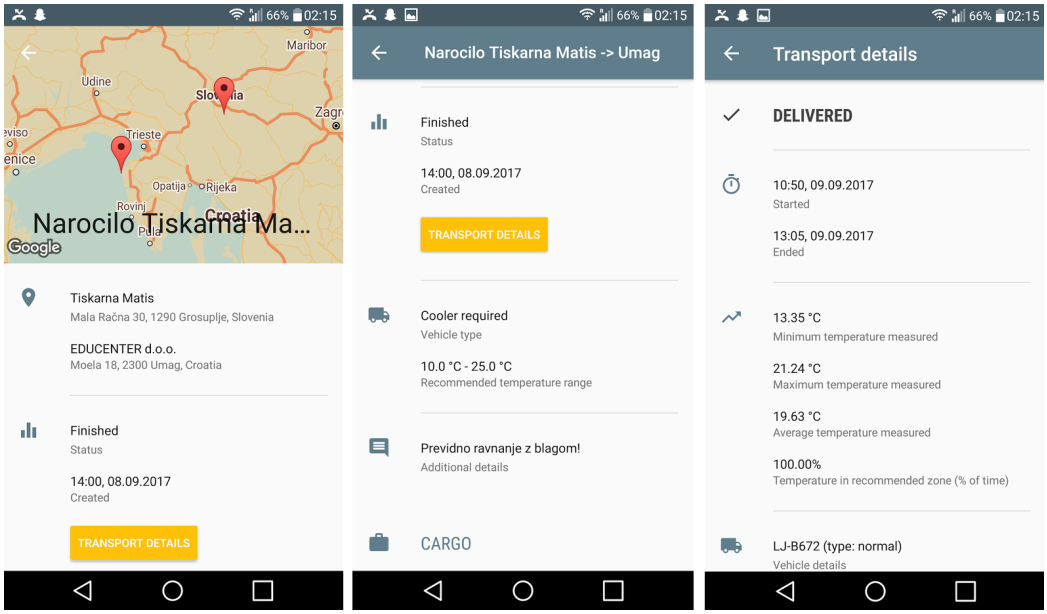


Slika 4.14: Dve aktivni naročili ob istem času.

mljevid vseh meritev in opozoril.

Oglejmo si prikaz rezultatov na simulaciji dveh transportov. Najprej bomo na lokaciji A (Grosuplje, Slovenija), prevzeli prvo naročilo (in začeli z zajemanjem meritev), ki ga bomo dostavili na lokacijo C (Umag, Hrvaška). Vmes se bomo ustavili še za prevzem drugega naročila na lokaciji B (Koper, Slovenija). Tudi tega bomo dostavili na lokacijo C (Umag, Hrvaška). Z oddajo obeh naročil se zajemanje meritev na lokaciji C ustavi.

Prvo naročilo je potekalo od vasi Mala Račna (blizu Grosuplja) pa do Umaga na Hrvaškem (Slika 4.15). Transport se je začel ob 10:50 in uspešno končal ob 13:05 z oddanim tovorom. Ves čas je potekal zajem meritev. Priporočeno območje za tovor naročila je od 10 °C do 25 °C. Iz podatkov o transportu lahko vidimo, da zajeta temperatura ni bila nikoli izven priporočenega območja. Najnižja izmerjena temperatura je bila 13.35 °C, najvišja 21.24 °C, povprečna pa 19.63 °C. Glede na to, da je bil uporabljen navaden kamion (brez hladilnika), lahko predpostavimo, da je bila med transportom tempe-

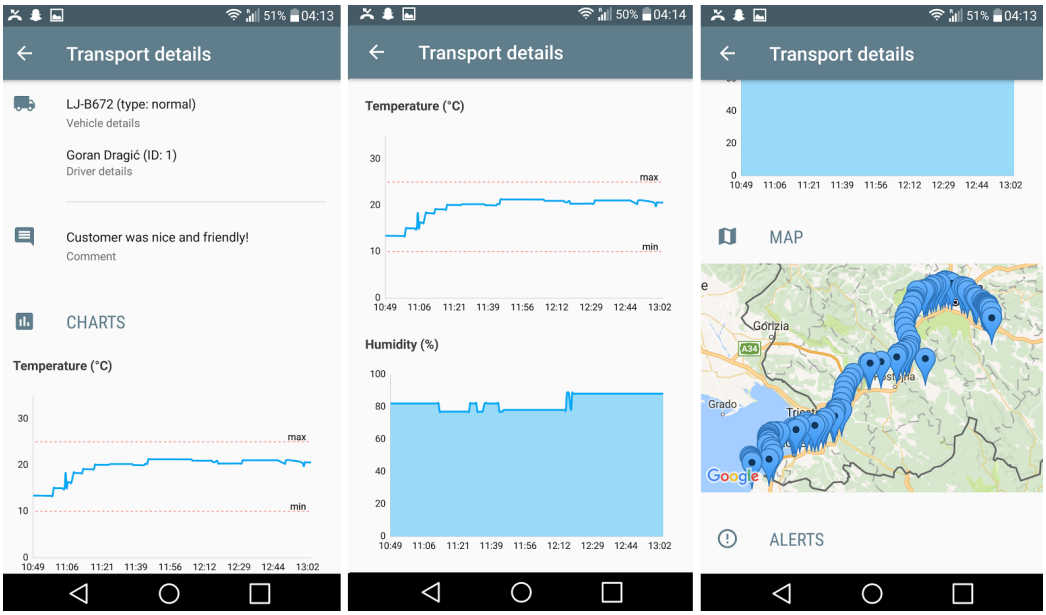


Slika 4.15: Informacije o naročilu (levo in v sredini) ter analiza zajetih meritev (desno).

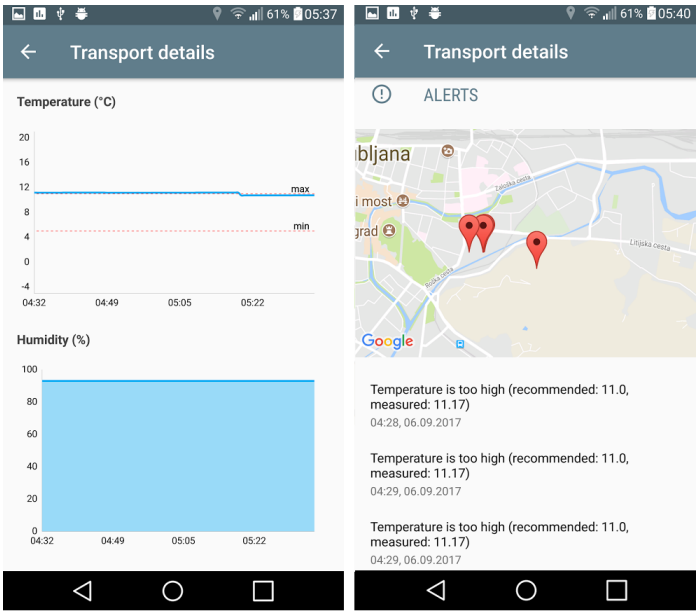
ratura v prostoru za tovor podobna izmerjeni temperaturi ozračja.

Na sliki 4.16 (levo) lahko vidimo še podatke o vozniku, ki je opravil transport. Sledita grafa temperature in vlage skozi čas (Slika 4.16, v sredini in desno). Pri grafu temperature sta dodani dve rdeči prekinjeni črti, ki označujeta spodnjo in zgornjo mejo priporočenega območja, tako da je jasno razvidno, če je bila temperatura med transportom kdaj neustrezna (pri tem naročilu se to ni zgodilo). Pod grafi je prikazan zemljevid z lokacijami vseh meritev. Vidimo, da so točke modre; v kolikor izmerjena temperatura na neki lokacija ne bi bila ustrezna, bi na tistem mestu opazili rdečo točko (primer na sliki 4.17).

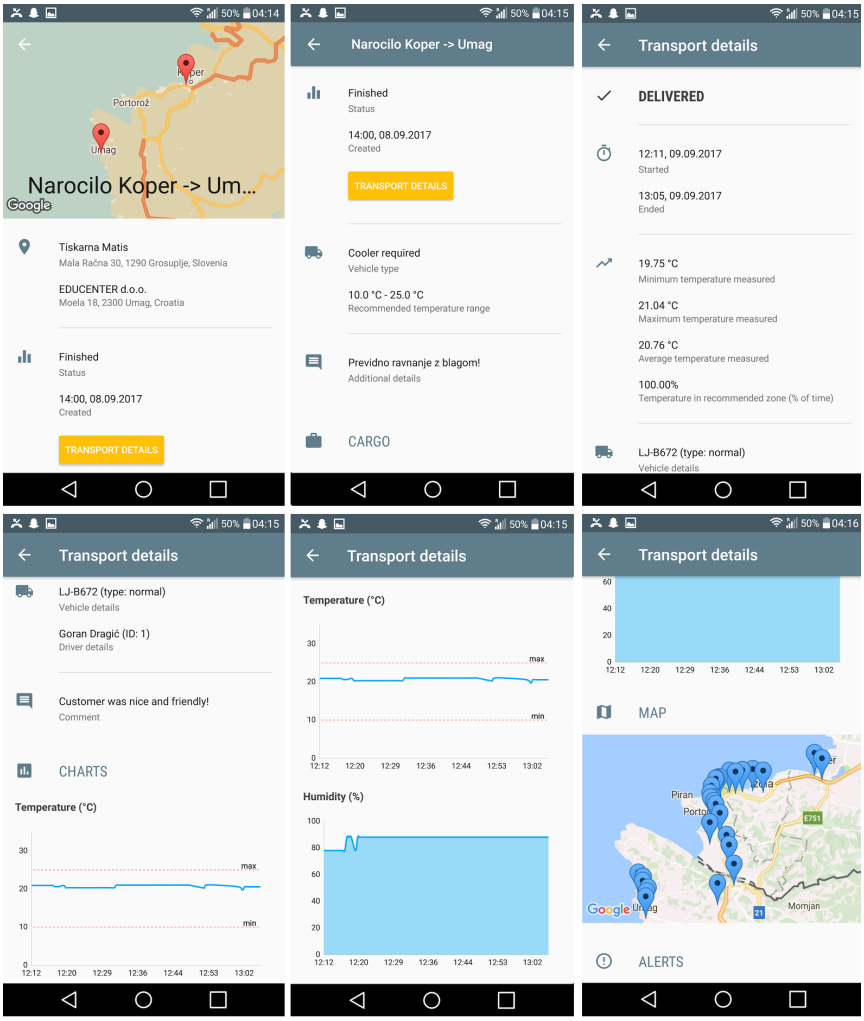
Med transportom prvega naročila smo v Kopru prevzeli še eno naročilo. Obe naročili sta imeli isto končno lokacijo: mesto Umag na Hrvaškem. Po predaji obeh tovorov in ponovnem branju QR kod obeh naročil, se je zaje-



Slika 4.16: Grafa temperature (levo) in vlage (v sredini) ter zemljevid (desno).



Slika 4.17: Primer transporta, kjer je izmerjena temperatura nekaj časa izven priporočenega območja - temperatura gre čez zgornjo rdečo prekinjeno črto (levo); na zemljevidu so te meritve označene z rdečimi točkami (desno).



Slika 4.18: Informacije o drugem naročilu (Koper - Umag).

manje meritev ustavilo. Vsa naročila so bila uspešno dostavljena. Na sliki 4.18 lahko vidimo zajete podatke o transportu drugega naročila. Povprečna temperatura je nekoliko višja (20.76 °C), vseeno pa so vse zajete meritve znotraj priporočenega območja. Tudi pri tem transportu velja, da so meritve temperature uporabne za analizo, saj odražajo realno stanje v prostoru za tovor (uporabljen isti kamion kot pri prvem naročilu!).

Poglavje 5

Sklepne ugotovitve

V diplomskem delu smo se spoznali s problemom transporta živil, ki je za številna, predvsem pa manjša podjetja, zahteven izziv. Čeprav je na trgu že veliko obstoječih rešitev, pa je prostora za napredek še vedno dovolj.

Odločili smo se za izdelavo mobilnega nadzora transporta živil, ki daje poudarek na ažurnem in enostavnem spremljanju prevoza tovora. Aplikacija deluje v povezavi z informacijskim sistemom prevozniškega podjetja. Z REST klici na strežnik pridobiva iz podatkovne baze podatke o uporabnikih, vozilih in naročilih.

Za čas celotnega transporta s pomočjo mobilne naprave aplikacija zajema GPS koordinate, na podlagi katerih s spletne storitve dobimo podatke o temperaturi in vlagi ozračja. Te nato primerjamo s podatki o tovoru, ki smo jih prenesli s strežnika podjetja in v primeru odstopanj od priporočenih vrednosti takoj opozorimo uporabnika. Meritve temperature ozračja so ustrezne pri transportih, kjer se ne uporabljajo hladilne naprave. V takih primerih lahko predpostavimo, da podatki iz vremenske postaje ustrezajo dejanskim pogojem v prostoru za tovor kamiona.

Z implementiranimi osnovnimi funkcionalnostmi - zapisom vseh doga-

janj celotnega transporta pri prevozniku, analizo kritičnih točk v verigi in sporočanjem neustreznih dogodkov - smo zadostili našim prvotnim ciljem: boljšim, bolj kontroliranim in bolj transparentnim transportom živil. Prevoznik lahko kar s svojo mobilno napravo, ki jo ima vedno pri roki, spremlja stanje tovora, hkrati pa ga aplikacija sama opozarja na nepravilnosti.

Prevoznik ima podatke o celotnem transportu in ob zaključku se ti podatki prenesejo tudi na strežnik podjetja. S svojo mobilno napravo lahko že ob predaji tovora pokaže ustrezne rezultate, povezane z zahtevami v naročilu. Gre za rešitev, ki je obojestransko koristna. Če je opravljena storitev dobra, bo prevoznik z veseljem pokazal dobre rezultate. V nasprotnem primeru, ko je tovor pokvarjen ali poškodovan, pa ima tudi naročnik vse informacije, da lahko izve, ali je za težave kriv prevoznik ali kdo drug.

5.1 Možne izboljšave

Seveda ima naša rešitev tudi pomanjkljivosti; testirana je bila na eni sami Android napravi, torej lahko na drugih napravah pričakujemo obilico težav. Potrebno bi bilo resno testiranje na čim več napravah in odpravljanje hroščev. Izvedeni sta bili le dve simulaciji transporta. Vseeno pa je največja slabost naše rešitve ta, da ne dobiva podatkov o temperaturi in vlagi s senzorjev v prtljažnem prostoru tovornjaka, temveč za to uporabljamo spletno storitev.

Čeprav je veliko prostora za napredek že pri obstoječih funkcionalnostih, pa imamo vseeno v mislih tudi že nekaj novih nadgradenj:

- pošiljanje podatkov na strežnik ob vsakem zajemu meritev
- algoritem, ki bi predlagal optimalen vrstni red izvedbe naročil
- pridobivanje podatkov o temperaturi in vlagi s senzorja v notranjosti prostora za tovor

- boljši uporabniški vmesnik
- optimizacija za najnovejšo različico Androida (8.0 *Oreo*)

5.2 Spremna misel

Temo diplomske naloge sem si izbral z namenom naučiti se programirati mobilne aplikacije za operacijski sistem Android. Mislim, da mi je v veliki meri uspelo. Čeprav je bilo na trenutke naporno in težko, pa sem se ob razvoju ogromno naučil. In četudi je za vse druge to le še ena številka v neskončni vrsti mobilnih aplikacij, pa je to moja prva.

Upam in verjamem, da vsekakor ne zadnja.

Literatura

[1] Android developers. Dosegljivo: https://developer.android.com/about/dashboards/index.html. [Dostopano: 14.7.2017].

[2] Global positioning system history. Dosegljivo: https://www.nasa.gov/directorates/heo/scan/communications/policy/GPS_History.html. [Dostopano: 10.8.2017].

[3] Introduction to mobile vision. Dosegljivo: https://developers.google.com/vision/introduction. [Dostopano: 30.7.2017].

[4] Overview of google play services. Dosegljivo: https://developers.google.com/android/guides/overview. [Dostopano: 30.7.2017].

[5] Road freight transport by type of goods. Dosegljivo: http://ec.europa.eu/eurostat/statistics-explained/index.php/Road_freight_transport_by_type_of_goods#Road_freight_transport_by_type_of_goods_.28NST_classifications.29. [Dostopano: 10.8.2017].

[6] Support library features guide. Dosegljivo: https://developer.android.com/topic/libraries/support-library/features.html. [Dostopano: 14.7.2017].

[7] Web services glossary. Dosegljivo: https://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/#webservice. [Dostopano: 13.8.2017].

[8] Myo Min Aung and Yoon Seok Chang. Temperature management for the quality assurance of a perishable food supply chain. *Food Control*, 40, 2014.

[9] Myo Min Aung and Yoon Seok Chang. Traceability in a food supply chain: Safety and quality perspectives. *Food Control*, 39, 2014.

[10] Jenny Gustavsson, Christel Cederberg, Ulf Sonesson, Robert van Otterdijk, and Alexandre Meybeck. Global food losses and food waste: extent, causes and prevention. fao, rome, 2011.

[11] Reiner Jedermann, Thomas Pötsch, and Chanaka Lloyd. Communication techniques and challenges for wireless food quality monitoring. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 372(2017), 2014.

[12] Hilton Charles E. (Eds.) Meldrum Jeff. From biped to strider: The emergence of modern human walking, running, and resource transport, 2004.